Using 2D Hierarchical Heavy Hitters to Investigate Binary Relationships

Trivellato, Daniel ; Mazeika, Arturas ; Böhlen, Michael Hanspeter

Abstract: This chapter presents VHHH: a visual data mining tool to compute and investigate hierarchical heavy hitters (HHHs) for two-dimensional data. VHHH computes the HHHs for a two-dimensional categorical dataset and a given threshold, and visualizes the HHHs in the three dimensional space. The chapter evaluates VHHH on synthetic and real world data, provides an interpretation alphabet, and identifies common visualization patterns of HHHs.

DOI: https://doi.org/10.1007/978-3-540-71080-6_14

Using 2D Hierarchical Heavy Hitters to Investigate Binary Relationships

Daniel Trivellato, Arturas Mazeika, and Michael H. Böhlen

Faculty of Computer Science, Free University of Bozen-Bolzano,
Piazza Domenicani 3, 39100 Bolzano, Italy
{datrivellato,arturas,boehlen}@inf.unibz.it

Abstract. This paper presents VHHH: a visual data mining tool to compute and investigate hierarchical heavy hitters (HHHs) for two-dimensional data. VHHH computes the HHHs for a two-dimensional categorical dataset and a given threshold, and visualizes the HHHs in the three dimensional space. The paper evaluates VHHH on synthetic and real world data, provides an interpretation alphabet, and identifies common visualization patterns of HHHs.

Key words: Visual Data Mining, Hierarchical Heavy Hitters, Lattice Structure, Ordering of Categorical Data

1 Introduction

The amount of data stored in todays databases continuously increases due to advances in automatic data gathering techniques, growth of the Internet, and the increasing popularity of E-commerce. The analysis of this data with the purpose of anomaly detection, overview of the data is of great importance.

This paper presents *VHHH* (Visualization of Hierarchical Heavy Hitters): a tool to visually investigate binary relations. VHHH computes a lattice summary structure for the data, and allows to detect anomalies and get an overview of the data.

The input of the VHHH method is a two dimensional dataset with hierarchical attribute values and a threshold. Hierarchical attribute values are specified explicitly, e.g., Italy→Veneto→Verona or a value hierarchy can be specified along with atomic attributes values. Hierarchical attribute values are equivalent to atomic attribute values together with a hierarchy and we use them interchangeably in this paper. VHHH computes the hierarchical heavy hitters (HHHs) of a dataset and visualizes the HHH information in three-dimensional space. Intuitively, a heavy hitter is a frequent item in the dataset. Hierarchical heavy hitters extend heavy hitters with the grouping of data according to a hierarchy. For example, if tuple Italy→Veneto→Verona is not a heavy hitter, it is grouped with all non heavy hitter tuples in Veneto, and the grouped count of Italy→Veneto is checked against the given threshold.

The application area of our solution is the analysis of binary relations captured by two-dimensional datasets. Examples include data-flow datasets with

"Source" and "Destination" attributes such as Internet router traffic datasets (traffic from source IP to destination IP), firewall data (IP address accesses specific port in the computer), airline database (passengers travel from an arrival to a destination airport), etc. Other application areas are trend and pattern analysis over two attributes of a dataset (for example, analysis of the relationship between car manufacturers and broken parts in cars) and geographical information systems.

A straightforward solution to analyze HHH is to directly visualize a lattice to the observer, and show the heavy hitters for each node (cf. Figure 7). This shows the lattice structure and allows to identify specific HHHs. However, the analysis is limited to the identification of specific HHHs. The analysis of vertical relationships (e.g., Italy is a part of Europe) and horizontal relationships (both Italy and Germany are European countries) between HHHs is not supported for 2D data and the investigation of trends and patterns of HHHs is difficult. VHHH not only allows to identify specific HHHs in the dataset, but also investigate vertical and horizontal relations as well as understand the patterns and trends between HHHs.

The paper makes the following contributions:

- We developed VHHH, an interactive visualization technique for the visualization and analysis of HHHs of two-dimensional datasets. The solution computes two-dimensional HHHs, and visualizes the HHHs as rectangles in the three dimensional space. The HHHs are visualized in three-dimensional space with two attributes mapped to the X and Y axes, respectively and the grouping level mapped to the Z axis.
- We extend the HHH method [7] and contribute with a new count-balancing technique in the computation of HHHs. As a result, the computation of heavy hitters is not only correct at the bottom and top level of the lattice, but also more accurate at intermediate levels of the lattice.
- We propose two orderings of categorical values based on HHH information: dataset ordering and HHH ordering. With dataset ordering the categorical values are ordered according to their frequency in the dataset. With HHH ordering, the categorical values are ordered according to the frequency of HHHs. Both orderings preserve the hierarchical information of the attributes.
- We propose a VHHH interpretation alphabet to mine datasets and identify common visualization patterns. We give an extensive experimental evaluation of VHHH on synthetic and real world datasets. Our analysis confirms that VHHH helps to detect anomalies and provides an overview of the data.

The paper is organized as follows. In Section 2 we review related work. Section 3 explains the computation of HHH as well as our count-balancing rule for the HHH method. Section 4 presents our VHHH solution. Section 5 evaluates VHHH experimentally, provides an interpretation alphabet for VHHH, and investigates typical patterns. Conclusions and future work are offered in Section 6.

2 Related Work

We review related work in four main areas: (i) HHHs, (ii) statistical summaries, (iii) visualization of categorical data, and (iv) ordering of categorical data.

In the area of HHHs the main issues have been the memory and time efficient computation of HHHs [9, 16, 18] and the computation of HHHs for streaming data [6, 7]. In this paper we focus on the data analysis potential of HHHs. We extend the computation of HHHs [7] with the count-balancing rule in order to have more accurate comparison and evaluation of HHHs at intermediate levels of the lattice. Hershberger et al. [9] propose a method to visualize the lattice of HHHs, but they focus on the computation of HHHs information and the data mining potential of HHHs is not considered.

The grouping of data and presenting summaries of the data is a wide research area. For example, data warehousing [14] groups the data according to the lattice into a high dimensional data cube and uses the cube for fast query answering. Simple visualization methods of the groupings offer the possibility to roll-up/down selected groupings according to selected attributes and visualize the data as scatter-plots. Association rules [1] identify large itemsets with the help of a lattice. Different visualizations have been proposed for association rules [4, 17, 12]. The main issue is the information visualization itself and the interpretation of the results is not worked out and left to the data analysts.

Visualization techniques for categorical data [2, 8, 5, 11] focus on information visualization and a comprehensive presentation of all properties of the data including bar-charts, pie-charts, parallel sets, histograms, tree-maps, and pair-maps. The methods focus on the information presentation aspects of the categorical data. In this paper we are focus on the visualization of hierarchical heavy hitters with the purpose of data analysis. The goals of the developed tools are quite different: the visualizations should not be necessarily nice, but informative.

Any visualization using a Cartesian coordinate system requires an ordering of data. For categorical data no obvious ordering is available and various authors have studied effective ordering techniques for categorical data [13, 3, 10, 15]. Ma and Hellerstein [13] identify two key tasks: (i) grouping similar values for a single categorical attribute and (ii) relate values of different attributes. For hierarchical heavy hitters, task (i) is provided by the hierarchies of attributes, and task (ii) is determined by the co-occurrence of values in the data. To order values of a single categorical attribute Chen et al. [10] proposes a mapping technique from categorical values to numbers based on a distance matrix. Similarly, Rosario et al. [15] present a technique to map nominal values to numbers to determine an ordering and distances. In our approach, attribute values are mapped to numbers according to their frequency in the dataset.

3 Two-Dimensional Hierarchical Heavy Hitters

Two-dimensional HHH are based on three concepts: (i) *filtering*, (ii) *grouping*, and (iii) *distribution of counts*. We explain these concepts in turn. Section 3.1

explains the concept of filtering for one-dimensional heavy hitters. Section 3.2 explains the concept of grouping for one-dimensional heavy hitters with hierarchies. Finally, Sections 3.3 and 3.4 explain the concept of the distribution of counts for two-dimensional hierarchical heavy hitters.

3.1 Filtering

One-dimensional heavy hitters filter according to a predefined threshold: all tuples with a value above a given threshold are returned as heavy hitters.

1D heavy hitters can be expressed with an SQL select query. An example of such an SQL query together with a dataset are given in Figure 1. The SQL query determines the heavy hitters for the given database and threshold. In our example the heavy hitters for threshold 30,000 are Bill and Jack. Intuitively, these two clients are the profitable clients.

```
SELECT Sales.Name
FROM Sales
WHERE Sales.Amount > 30,000
```

(a) Select Query

Name	Amount
Bill	50,000
John	24,500
Mark	26,000
Jack	35,000

(b) Dataset

Fig. 1. 1D HH Example for Sales Data

The computation of heavy hitters requires a dataset and a threshold as input. If the input dataset consists of at least one numerical attribute the numerical attribute can be used to filter the dataset. If the database does not include a numerical attribute, the number of occurrences (count) can be used to filter the database. An example is web log of Internet addresses accessed by a computer. In this case the heavy hitters represent the web sites visited most frequently by the computer.

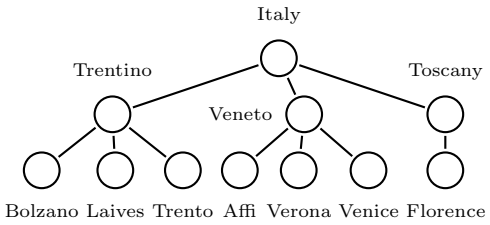
3.2 Grouping 1D Hierarchical Data

One-dimensional HHHs extend heavy hitters with the concept of grouping according to a hierarchy. The computation of HHH requires a database, the attribute value hierarchy, and a threshold as input. HHHs are computed iteratively. First we filter the database by deleting the HHHs and the tuples that contributed to the HHHs. The remaining tuples are grouped according to the hierarchy and further filtered. The process is repeated until no more grouping is possible or all tuples have been removed.

Consider threshold 50,000 and the data and hierarchy for attribute Municipality in Figure 2. The database records total sales amounts in individual

Municipality	Amount
Bolzano	60,000
Laives	30,000
Trento	70,000
Affi	20,000
Verona	55,000
Venice	40,000
Florence	40,000

(a) Database



(b) Hierarchy for Municipality Attribute

Fig. 2. Sales Data and the Hierarchy Information of Italian Municipalities

municipalities in Italy (cf. Figure 2(a)). The municipalities are grouped into regions and the regions are grouped into the root group Italy (cf. Figure 2(b)).

The computation of HHHs is an iterative process of filtering and grouping steps (cf. Figure 3). The first filtering step returns Bolzano, Trento, and Verona as HHHs. These tuples are removed (cf. Figure 3(a)). Next the tuples are grouped according to the given hierarchy: Laives is grouped into Trentino-South Tyrol, Affi and Venice into Veneto, and Florence into Toscany (cf. Figure 3(b)). This completes the first iteration. In the second iteration, the filtering outputs Veneto as a heavy hitter (cf. Figure 3(c)) and the grouping groups the remaining tuples into Italy (cf. Figure 3(d)). In the last iteration Italy is added as a HHH. All HHHs of the data are shown in Figure 3(e).

Municipality	Amount
Laives	30,000
Affi	20,000
Venice	40,000
Florence	40,000

(a) Step1, Filter

Group	Amount
Trentino-South Tyrol	30,000
Veneto	60,000
Toscany	40,000

(b) Step1, Group

Group	Amount
Trentino-South Tyrol	30,000
Toscany	40,000

(c) Step2, Filter

Group	Amount
Italy	70,000

(d) Step2, Group

Group	Amount
Bolzano	60,000
Trento	70,000
Verona	55,000
Veneto	60,000
Italy	70,000

(e) HHHs

Fig. 3. Step by Step Computation of 1D HHHs

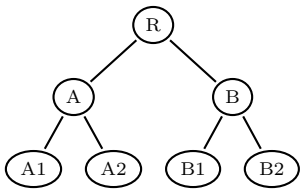
3.3 Grouping 2D Hierarchical Data

For the computation of HHH for two-dimensional data we have to generalize the hierarchy to a lattice. The lattice is the result of grouping data according to two independent hierarchies.

Consider the flight database in Figure 4. The database records the departure airport, the arrival airport, and the number of daily flights from the departure to the arrival airport. Grouping can be done according to the hierarchy of the departure airport (cf. the left branch in Figure 7(b)), the hierarchy of the arrival airport (cf. the right branch in Figure 7(b)), or both hierarchies (cf. the center part in Figure 7(b)). Note that in this example the two hierarchies are identical. In general, the hierarchies may be different.

Departure	**Arrival**	**No.Flights**
$R \to A \to A1$	$R \to B \to B1$	2
$R \to A \to A2$	$R \to B \to B1$	4
$R \to A \to A1$	$R \to B \to B2$	1
$R \to A \to A2$	$R \to B \to B2$	1
$R \to B \to B1$	$R \to A \to A1$	3
$R \to B \to B1$	$R \to A \to A2$	3
$R \to B \to B2$	$R \to A \to A1$	1
$R \to B \to B2$	$R \to A \to A2$	2
$R \to B \to B2$	$R \to B \to B1$	1

(a) The Dataset



(b) Hierarchy for Departure and Arrival Attributes

Fig. 4. Input for 2D HHHs

The combination of the hierarchies yields a lattice. The lattice is organized into layers and each node in the lattice is at a particular height from the bottom of the lattice. The height of the node denotes the shortest path to the node from a leaf. The height of a leaf node is 0. At the first level the minimal path is 1. This is the result of grouping according to the departure (1+0=1), or according to the arrival (0+1=1) airport. At the second level the shortest path may be gotten by grouping twice according to the departure airport (2+0), twice according to the arrival airport (0+2=2), or once according to the departure and once according to the arrival airport (1+1=2).

3.4 Computation of Counts in a Lattice

The computation of counts of grouped data is more complicated in the two-dimensional case. In contrast to trees each node in a two-dimensional lattice can receive data from nodes that summarize partially overlapping data. For example, node 2.2 in Figure 7 groups the data in the database (node 0.1) by grouping data from nodes 1.1 and 1.2. Tuples that contribute to node 1.1 and node 1.2 in node 2.2 will be counted twice. There are different approaches to handle this problem,

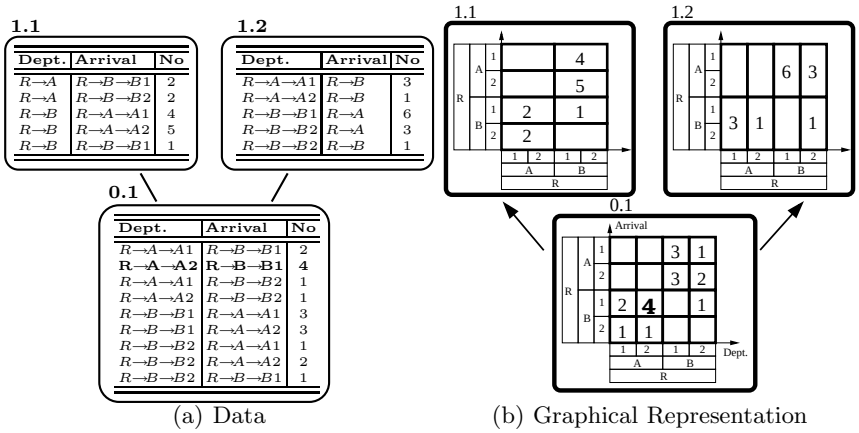
including proportional distribution and split of counts. Our computation of two-dimensional HHH is based on the count balancing rule, which combines the overlap and split rule [7]. As a result, the computation of HHHs is not only correct at the lowest and the top levels of the lattice but also is more accurate for intermediate levels of the lattice.

The computation of counts in the lattice is done iteratively from the lowest level to the root of the lattice. In each iteration the counts are computed for one level of the lattice. One iteration consists of three steps: (i) grouping of counts from the lower level, (ii) filtering the counts for HHHs, and (iii) balancing the counts (ensuring that the same database tuple is not counted twice). We illustrate the steps by an example and show how to compute the first iteration (compute the HHH information for nodes 1.1 and 1.2).

The *grouping* step gets the tuples from the children and groups the counts according to the attribute value hierarchy.

Getting the counts is straightforward: tuples have to be retrieved from one of the children only. If a node has more than one child (for example node 2.2 in Figure 7) then the counts can be retrieved from an arbitrary child of the node, since all children summarize the exact same tuples (for example node 1.1 and node 1.2 in Figure 7).

The grouping of the counts according to the level of an attribute selects the level for an attribute and sums up the counts according to the given level. Consider Figure 5. The grouping up to the 0th level for attribute "Departure" selects all distinct values in the column. These are $R \rightarrow A \rightarrow A1$, $R \rightarrow A \rightarrow A2$, $R \rightarrow B \rightarrow B1$, and $R \rightarrow B \rightarrow B2$ for node 0.1. The grouping up to the 1st level for the "Departure" selects all distinct values ignoring most detail leaf node information. This results in two values: $R \rightarrow A$ and $R \rightarrow B$. Similarly, the grouping up to the 2nd level returns record R.

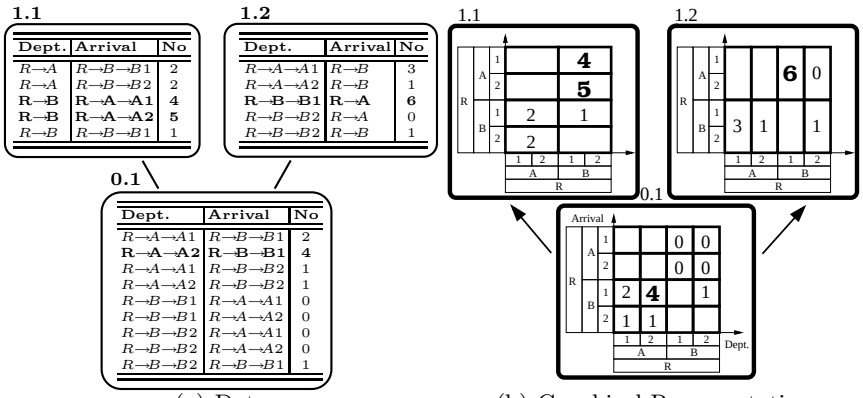


(a) Data (b) Graphical Representation

Fig. 5. Grouping of Counts at Height 1 in the Lattice

For node 1.1 the grouping step aggregates the flight numbers for the "Departure" attribute up to the 1st level, up to the 0th level for the attribute "Arrival". Similarly, the grouping step for node 1.2 aggregates the flight numbers for the "Departure" attribute up to the 0th level and for the "Arrival" attribute up to the 1st level. The result of this step is illustrated in Figure 5.

The *filtering* step scans the aggregated counts and identifies hierarchical heavy hitters (cf. tuples in bold in Figure 6).



(a) Data (b) Graphical Representation

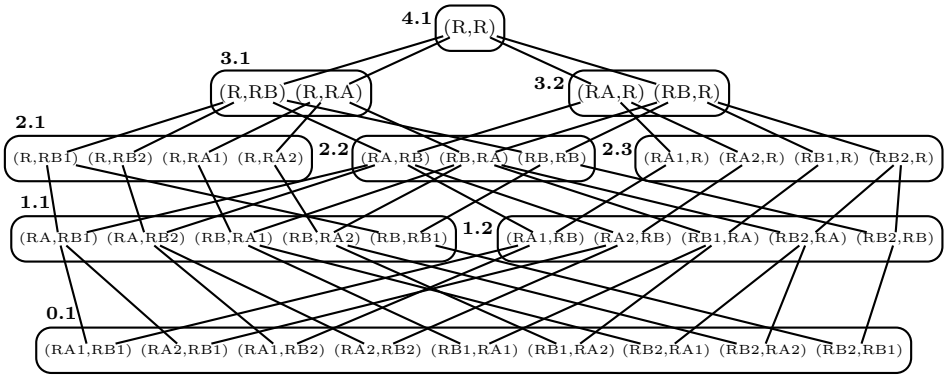
Fig. 6. Balancing of Counts at Height 1 in the Lattice

The *balancing* step guarantees that no database tuple is counted twice in any node in the lattice. Technically, this is done in two steps. First, we scan each node (nodes 1.1 and 1.2 in Figure 6). For each node we scan the HHHs (cf. $(\mathbf{R} \rightarrow \mathbf{B}, \mathbf{R} \rightarrow \mathbf{A} \rightarrow \mathbf{A1}, \mathbf{4})$ and $(\mathbf{R} \rightarrow \mathbf{B}, \mathbf{R} \rightarrow \mathbf{A} \rightarrow \mathbf{A2}, \mathbf{5})$ in node 1.1, $(\mathbf{R} \rightarrow \mathbf{B} \rightarrow \mathbf{B1}, \mathbf{R} \rightarrow \mathbf{A}, \mathbf{6})$ in node 1.2). For each HHH we decrease the count of the contributing tuples in the children nodes. After this step some of the counts may be 0 or even negative. In case the counts are negative, we set their count to 0 (cf. node 0.1 Figure 6). After this step, the contributing tuples of children nodes (node 0.1) will all have count 0.

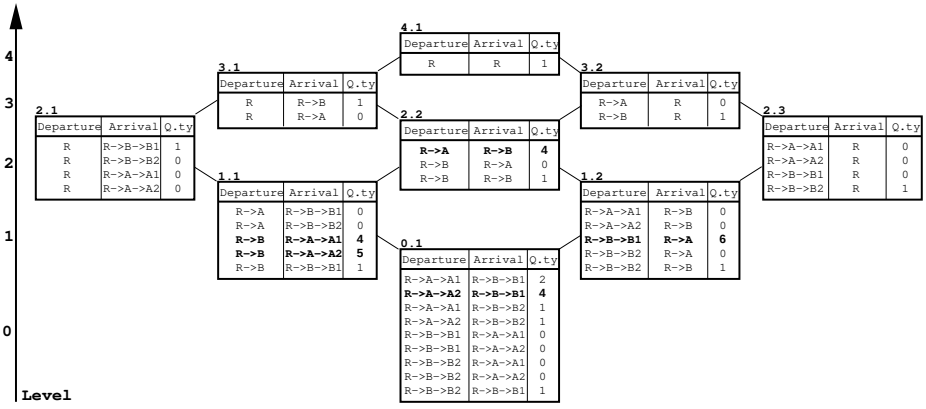
Second, the balancing step recomputes the counts of non-HHHs on the respective level (nodes 1.1 and 1.2). This ensures that the data distribution of non-HHH in the nodes is exactly the same (the total count of non-HHH in 1.1 is $2 + 2 + 1 = 5$ and the total count for non-HHH in 1.2 is $3 + 1 + 1 = 5$). This finishes the first iteration. Independent whether further counts are propagated from node 1.1, or node 1.2, the result of computation of counts in node 2.2 is the same (cf. node 2.2 in Figure 7(b)).

Figure 7 shows the complete result of the computation of the counts. Note, that in the first iteration the grouping step processes the leaf node 0.1. During other iterations (e.g., iteration 2) the grouping step can involve the grouping of counts from two inputs (for example for node 2.2 the input may be either 1.1

or 1.2). In this case the grouping step can select the input node (either 1.1 or 1.2) arbitrarily. The consistency of the result is ensured by the balancing step: both nodes summarize exactly the same information, and therefore the result does not change if the left or the right node is used as input.



(a) Attribute Value Hierarchy. For Simplicity Labels of the Nodes are Shortened: for Example (R→A→A1, R→B→B1) is Labeled as (RA1, RB1)



(b) The Lattice. Individual Links Between the Nodes are Abstracted and only the Relations Between the Levels are Shown

Fig. 7. The Lattice of the Flight Dataset (cf. Figure 4)

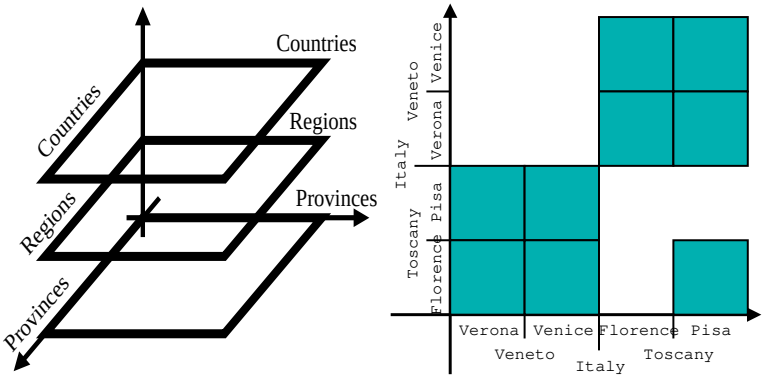
4 VHHH: Visualization of Hierarchical Heavy Hitters

VHHH, the visualization tool for two-dimensional HHHs, is based on two concepts: (i) visualization of the lattice together with heavy hitter information and (ii) ordering of the categorical data along the axis. We discuss the solutions in

turn in Sections 4.1 and 4.2. In Section 4.1 we assume that the ordering of the categorical values is given and present the visualization of the lattice and the HHH information. In Section 4.2 we discuss the ordering of the categorical data.

4.1 Visualization of the Lattice and HHH Information

The following properties summarize the visualization of the lattice of two-dimensional HHHs (cf. Figure 8): (i) the lattice is visualized on planes parallel to the XY plane, (ii) level planes represent (multiple) lattice levels: the lowest level of the lattice is visualized on the XY (lowest) plane whereas the highest level is visualized at the top of the visualization, and (iii) hierarchical heavy hitters are visualized as rectangles on the plane.



(a) Level Planes (b) Visualization of HHH Information

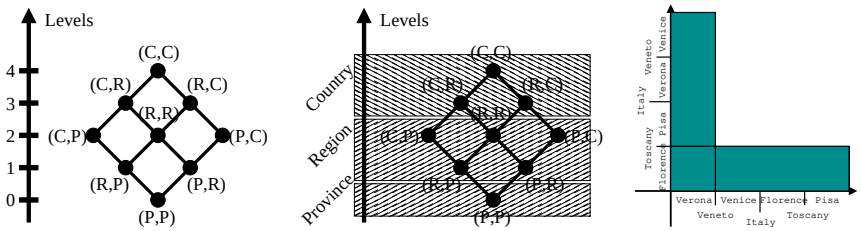
Fig. 8. VHHH: Visualization of Hierarchical Heavy Hitters

Figure 8 shows an example of VHHH. In Figure 8(a) the level planes of the lattice are shown. The lattice is visualized on three granularity levels: country level (Italy), region level (Veneto, Toscany), and province level (Verona, Venice, Florence, Pisa). An example of the visualization of HHH information on a plane is shown in Figure 8(b). Each rectangle in the visualization depicts a heavy hitter. For example the right most-bottom rectangle corresponds to heavy hitter (Pisa, Florence). In the visualizations of the implemented system, all HHHs are labeled. For example, the heavy hitter (Pisa, Florence) is labeled as Pisa/Florence.

The mapping of lattice levels to visualization planes is done in two ways: (i) the overview and (ii) the detailed view. In the detailed view all levels of the lattice are mapped to separate visualization planes. In this case, there are as many visualization planes as there are levels in the lattice (cf. Figure 9(a)).

In the overview view the number of levels is reduced and multiple levels are visualized on a single plane. The *overview granularity* allows to choose the

number of visualization planes. For example if the overview granularity is g then level x is visualized on plane $(x + 1)$ div g.



(a) Detail Mapping

(b) Overview Mapping, Overview Granularity $g = 2$

(c) Nodes May Overlap Even in the Detail View

Fig. 9. Mapping of the Lattice to the Visualization Planes. In Figures (a) and (b) C stands for Country, R for Region, and P for Province

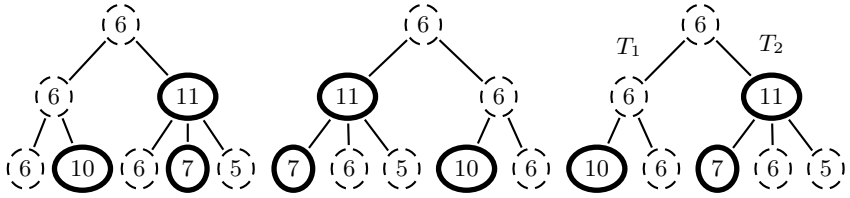
Note, that HHHs may overlap even in the detailed visualization of the lattice. This is because several nodes are visualized on the same level. An example of this case is illustrated in Figure 9(c). The HHH nodes are (Verona,Italy) and (Italy,Florence), which respectively are in the (Province,Country) and (Country,Province) groupings. Both groupings are on the second level of the lattice (cf. Figure 9(a)), and therefore are mapped to the same visualization plane.

4.2 Ordering of Categorical Data

VHHH uses the three-dimensional Cartesian coordinate system to visualize HHHs. Any visualization that uses a Cartesian coordinate system requires an ordering. For some categorical and hierarchical domains a natural ordering exists. An example of such a domain is time. The time domain is organized hierarchically (days are grouped into months, months into quarters, quarters into years, etc). Also, each level of the hierarchy comes with a full order (for example, June precedes July, the 1st quarter precedes the 2nd quarter, etc). For such datasets HHHs can be computed and visualized directly with the VHHH tool. For categorical attributes without ordering first an ordering should be established, and only then the data can be visualized with the VHHH tool. Examples of such attributes include fruits (apples neither precede nor succeed oranges), and countries. In this Section we establish orderings for categorical attributes without orderings.

We propose two strategies of ordering: (i) Hierarchical Heavy Hitter ordering (HHH ordering) and (ii) dataset ordering. Both orderings order the data according to the count information. HHH ordering considers only the count of HHH nodes, and aims at placing the largest HHHs at the beginning of the axis,

while the dataset ordering aims to allocate the largest leaf points at the beginning of the axis. We present our techniques for the one dimensional case. The generalization to two-dimensional orderings is straightforward.



(a) Heavy Hitter Information (b) HHH Ordering (c) Dataset Ordering

Fig. 10. Ordering of Categorical Data

Heavy Hitters Ordering. HHH ordering aims to position the largest HHHs at the beginning of the axis. Consider the HHH information in Figure 10(a) as an example (the threshold is 7). Since the largest HHH is with count 11 (at level 1), it is moved to the beginning of the axis. The same strategy is applied to order the tree according to the remaining HHHs. The ordering searches for the second largest HHH (count 10 on level 0), and moves it right after the 1st subtree. Finally, the HHH with count 7 is moved to the left most position of level 0. This completes the ordering, since there is no HHH node left that may be moved. The actual ordering of the database is obtained by scanning the leaves of the rearranged tree from left to right.

The complexity of each iteration of the algorithm is $O(s)+O(b)+O(h \cdot b)$, where s is the number of HHHs, b is the average branching factor of the tree, and h is the height of the HHH node from the root. The complexity depends on the following three possible cases of the algorithm: (i) search of the largest HHH L (complexity is $O(s)$ due to a linear scan), (ii) push L node to the left most unordered position on its level (complexity is $O(b)$), (iii) update of the path from L to root (complexity is $O(h \cdot b)$ due to the update of the coordinates of the nodes at all levels above the node L).

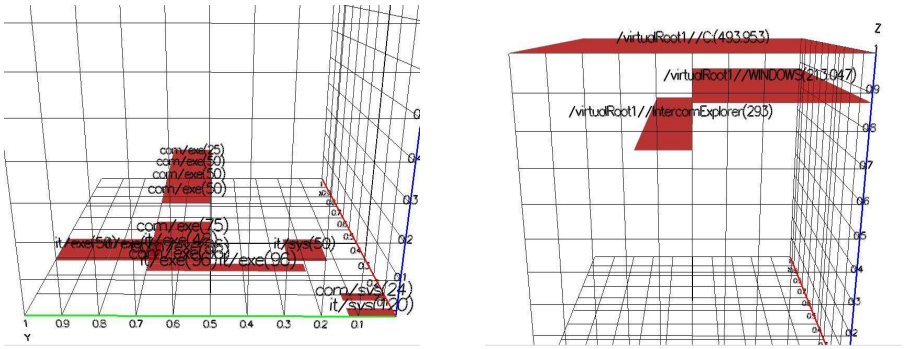
Dataset Ordering. Dataset ordering aims to position the data points according to their count. The dataset ordering for the dataset of Figure 10(a) is shown in Figure 10(c). First, the algorithm scans all leaves of the tree and identifies the largest leaf (the leaf with count 10). This determines the subtree that is positioned at the beginning of the axis (subtree T_1). The other trees that may change their positions as the process continues (subtree T_2). Second, the algorithm sorts subtree T_1 and determines the second largest leaf (the leaf with count 7). Finally, it sorts subtree T_2.

The complexity of an iteration of the algorithm is $O(l)+O(b \log b) + O(h \cdot b)$, where l is the number of leaves, b is the average branching factor, and h is the height of the tree hierarchy: the complexity of the search of the largest element (case (i)) is $O(l)$ (due to a linear scan of the leaves). The complexity of sorting the nodes of the current level (case (ii)) is $O(b \log b)$. The complexity of the update of the path from L to the root (case (iii)) is $O(h \cdot b)$.

5 Experiments

5.1 Case Studies with Real World Data

Firewall Logs. A firewall protects the computer from unsolicited access to and from the Internet, and it records all the Internet accesses of installed programs. High access rate of a particular program to a selected URL might point to a unsolicited spy-ware program accessing an untrusted web site and sharing private data. VHHH allows to quickly and easily detect and locate weak points in such a setting and identify the granularity of the problem.



(a) Personal Computer (b) Office Computer

Fig. 11. VHHH on Firewall Data

Figure 11 shows the VHHH of the firewall data. of a personal computer (Figure 11(a)) and an office computer (Figure 11(b)). The X axis maps the Internet address (for example `com`→`cnn`→`www`), the Y axis maps the computer program (`Programs`→`Mozilla`→`Firefox`) and the Z axis maps the grouping of the nodes of the lattice according to the hierarchical information.

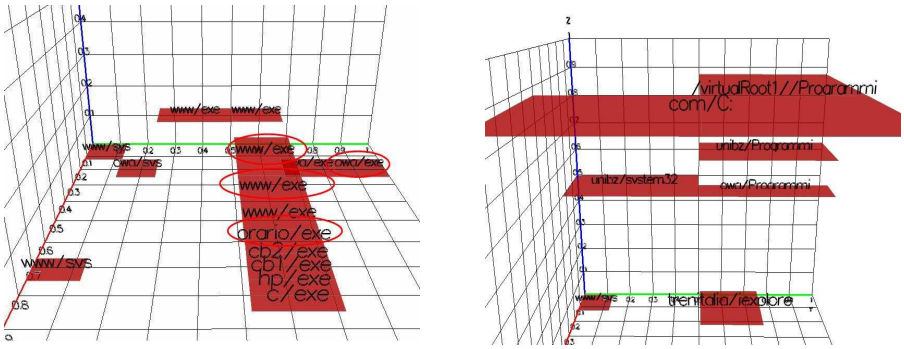
The HHHs for personal computers are quite different from the HHHs of the office computer. The HHHs of the personal computer are located at the bottom of the visualization, indicating very frequent accesses of particular programs/low level of groupings of programs and specific Internet addresses. This means that the personal computer might be infected, and might contain a number of viruses/spy-ware programs. A closer investigation of these specific Pro-

gram/Internet address pairs should be performed in order to understand potential weaknesses in the security of the personal computer. In contrast, the HHHs of the office computers are located at higher levels of granularity, and do not pose a security problem.

Similar patterns can be seen in the investigation of car (manufacturers, parts attributes) or (personnel, quality of work attributes) databases. For example, if the HHHs are located at the lower part of the visualization in (personnel, quality work) database, then the HHHs point to specific problematic workers in the company. Otherwise, if the HHHs located at the top then it shows that the problems lie in the management of the company and no particular worker causes the problem.

Figure 11 is an example where VHHH analyzes the data for a particular problem (infection of spy-ware). VHHH also allows to get an overview of the data and analyze data without an apriori goal or hypothesis. In this case, different threshold levels are visualized and shown to the analyst.

Figure 12 shows an overview of the data for two thresholds $t = 0.02$ and $t = 0.10$. The HHHs for the low threshold value (cf. Figure 12(a)) are more likely to be located at the bottom of the visualization (since very little grouping is required to reach the threshold), while the HHHs for high threshold values are located at the top of the of the visualization (cf. Figure 12(b)).



(a) Threshold 0.02 (b) Threshold 0.10

Fig. 12. Overview of Traffic Log Data

Figure 12(a) shows the HHHs for threshold $t = 0.02$. Two observations can be drawn from this visualization: (i) a few very specific programs are accessing a few very specific URLs (e.g., some `sys` files accessing `www` sites, or `exe` files accessing few different sites), and (ii) most of the URLs are accessed by executable programs (cf. right half of Figure 12(a) all HHHs are represented by `exe` programs). The latter is a trivial result once domain knowledge about the data is available (programs are grouped into executables). The specific `exe` programs

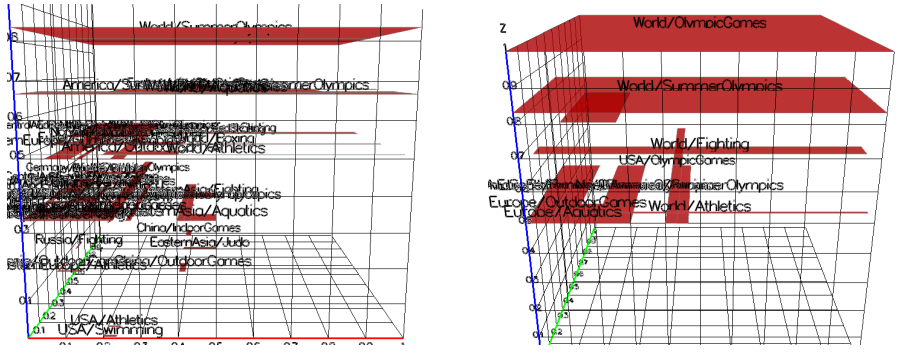
and URLs are MS Outlook and the mail server URL, and Internet Explorer and the URL of the Italian railways.

Figure 12(b) shows the HHHs for threshold $t = 0.10$. Similarly to Figure 12(a), there are a number of HHHs at the bottom levels of the visualization. These are the MS Outlook and Internet Explorer programs accessing the mail server and the Italian railways web site. These HHHs are so "heavy" that the increase of the threshold did not filter them out. The figure also shows that on the general level most of the programs and the system components are accessing the university domain (cf. (unibz, Programmi) and (unibz, system32) in Figure 12(b), just below the HHHs on the top level).

Olympic Games Gold Medals. This section investigates VHHH on gold medals won in each discipline of the Olympic games since year 2000 (the size of the database is approximately 800). The dataset is two-dimensional: the first dimension records information about the country (World→America→North America→USA for example) and the second dimension the sport (Olympic Games→Summer→Indoor→Table Tennis for example).

A number of interesting observations can be drawn from the VHHH analysis of the dataset (cf. Figure 13).

For threshold $t = 0.02$ (cf. Figure 13(a)) there are very few HHHs that are located at the bottom of the visualization. This shows that no specific country excels in some particular sport. Exceptions for this rule are (USA, swimming), (USA, athletics), (Russia, fighting), (China, indoor). These countries have long traditions in the selected sports and are large enough to have confident trends. Most of the HHHs are concentrated in the intermediate area with a lot of (continent, category) HHHs. At the top there are three big HHHs: (world, winter games), (world, summer games), (Africa, all).



(a) Threshold $t = 0.02$ (b) Threshold $t = 0.10$

Fig. 13. Gold Medals in Olympic Games

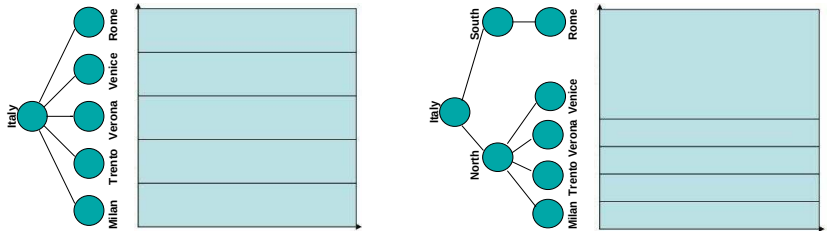
Figure 13(b) illustrates the HHHs for threshold $t = 0.10$. VHHH shows that there are much fewer HHHs for this threshold. One can see that there are a number of European medals in outdoor games and aquatics. There are HHHs on different groupings as well: on the world level (cf. (world,fighting) and (world,athletics) in Figure 13(b) , for example), on continent level (cf. (Europe,aquatics)), and single country (cf. (USA,all)). However, in general, HHHs are located either in the top grouping according to the country, or according to the game.

5.2 VHHH Alphabet

This section develops an interpretation alphabet for VHHH. The interpretation alphabet establishes interpretation primitives and allows the analyst to look for specific characteristics in visualizations, and provide a possible interpretation both on the analysis of the structure of the data (lattice) and the data (the meaning of the HHH information) for the visualizations.

The interpretation alphabet is based on the following characteristics: (i) node size, (ii) node shape, and (iii) node orientation. We review each of the characteristics in turn.

Node Size Information. The node size indicates parent-child relationship among nodes and the height of the node in the lattice hierarchy. For example, if a HHH is a superset of another HHH in a VHHH visualization, then the first node is an ancestor of the other. The size information allows to position the HHHs in the lattice. In general, smaller HHHs are located at lower levels of the lattice, and larger HHHs are located at the top of the lattice.



(a) (b) HHHs for Hierarchy in Figure 14(a) (c) (d) HHHs for Hierarchy in Figure 14(a)

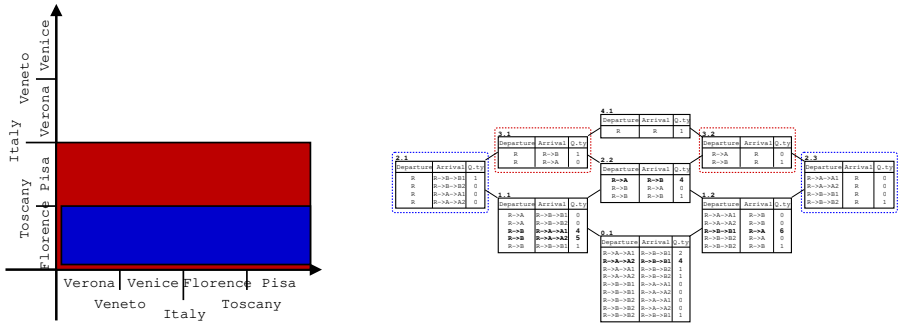
Fig. 14. Structure of the Lattice has a Great Impact on the Size of HHHs

Note that the structure of the attributes impacts the size of the HHHs. Consider, for example, the hierarchy illustrated in Figures 14(a) and 14(c). The visualization of all children of the tree structures is shown in Figures 14(b) and 14(d). Note, that there is a substantial difference in the size of the node

"Rome", since the introduction of an additional level in the hierarchy enlarged the size of the node for the visualization in Figure 14(d).

Node Shape. Shape information provides additional information about the nodes position in the lattice. While the node size determines the vertical position in the lattice, the shape information determines the horizontal position in the layer of the lattice.

The shape information determines the grouping to which a HHH node belongs. The more non-quadratic a node, the further it is located from the center of the lattice (cf. node (Italy, Florence) in Figure 15). The more quadratic a node, the more in the middle of the lattice is located (cf. node (Italy, Toscany) in Figure 15).



(a) Shape of the HHHs (b) Horizontal Position in the Lattice

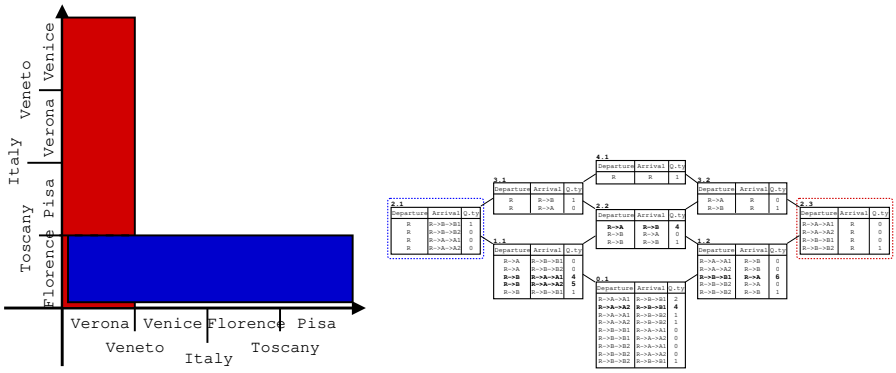
Fig. 15. Shape Information Provides the Horizontal Position of the Node in the Lattice

Node Orientation Information. Similar to shape information, orientation information also provides the horizontal position of the node in the lattice. The orientation of the node shows whether the node is located in the left, or in the right part of the lattice.

Figure 16 illustrates the orientation information. Nodes that are situated in the left part of the lattice are parallel to the X axis (cf. node (Italy, Florence) in Figure 16), while nodes that are situated in the right part of the lattice are parallel to the Y axis (cf. node (Verona, Italy) in Figure 16).

5.3 Pattern Investigation of VHHH

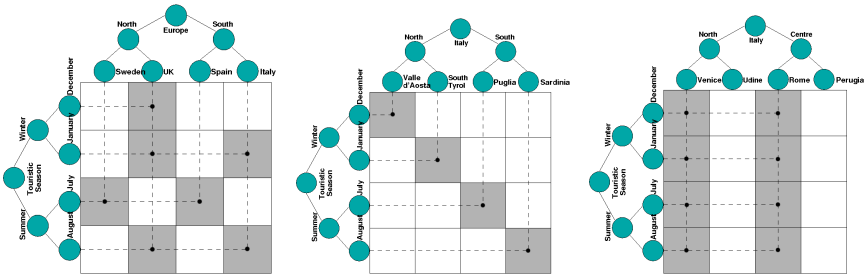
This section discusses typical HHH patterns that we found while mining real world datasets. In the following we describe the three most typical patterns: (i) chess board pattern (cf. Figure 17(a)), (ii) diagonal pattern (cf. Figure 17(b)), and (iii) flag pattern (cf. Figure 17(c)). These patterns can be formed on a single



(a) HHHs (b) Position in the Lattice

Fig. 16. The orientation of the node shows whether the node is located in the left, or in the right part of the lattice.

layer or on different layers of the lattice. For simplicity, we present the cases when the patterns are formed on the same level of the lattice.



(a) Chess Board (b) Diagonal (c) Flag

Fig. 17. Pattern Investigation of VHHH

Chess Board Pattern. Generally, in the chessboard pattern a HHH is followed by a non-HH node. Datasets with this pattern typically exhibit some sort of periodicity. A tourist agency database is an example, since the touristic concentration in European countries shrinks and grows during the months of the year (cf. Figure 17(a)). Note, that sometimes HHH nodes are grouped into sets. Intuitively, the size of chessboard squares varies according to the number of nodes in the set. For example, consider a scenario where most trips in the winter are organized towards north European countries, while in July and August tourism is concentrated in the south of Europe. This information would result in a visualization with two sets of HHHs, and hence two chessboard squares of

four leaf pairs each: a set including Northern countries in the winter months, and another represented by Southern countries combined with the two summer months.

Diagonal Patterns. In the diagonal pattern the HHHs form a diagonal line in the two dimensional plane (cf. Figure 17(b)). This is an indication of a gradual change as (for example) time increases.

An example of such a dataset is a tourist database of local regions of Italy. As the seasons of the year change, tourists change their preferences from going skiing in Valle d'Aosta and South Tyrol in December-January to summer vacations in Puglia and Sardinia during June-July.

Flag Patterns. In the flag pattern (cf. Figure 17(c)) stripes of HHHs are followed by stripes of non-HHHs. This pattern shows a periodicity for one attribute, and a steady behavior for the other one.

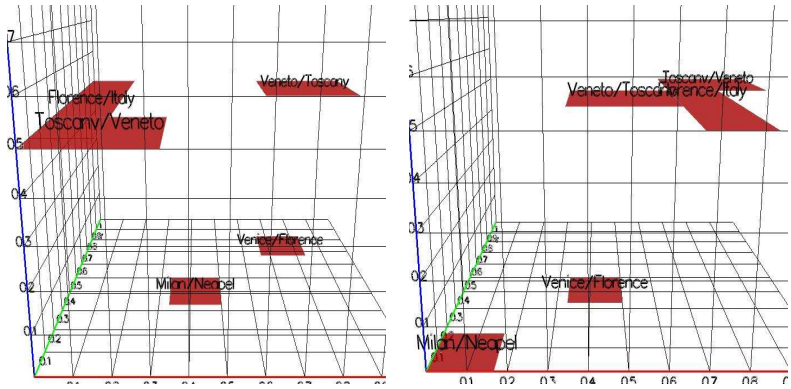
An example of such a database is a tourist database with time a attribute (for two winter and two summer months) and location attribute (four Italian cities from Rome, Venice, Udine, Perugia). VHHH shows that independently from the season a lot of tourists visit Venice and Rome. Also, not so much tourist concentration affects Udine and Perugia in any of the four months.

5.4 HHH Ordering versus Dataset Ordering

The ordering of data affects the position of data points on the X and Y axis. Consequently, different orderings imply a different positioning of nodes in the visualization. With HHH ordering, HHH nodes with the highest count are moved towards the origin of the XY plane. Dataset ordering moves the most frequent leaves to the beginning of the axis.

Figure 18 shows an example of VHHH with the two different orderings. We computed and visualized HHHs for a synthetic flight dataset with Italian departure and arrival airports. The lattice and groupings of the investigated dataset are identical to the dataset presented in Sections 4.1 and 4.2 (Figures 8 and 9). However, dataset elements are slightly different.

Assume, that the National Flight Authority (NFA) discovers that delays in HHH routes are due to the inefficiency of boarding procedures, and decides to experiment a new technique that is predicted to improve those processes. The new procedure can not be introduced in all airports simultaneously, because its reliability is not guaranteed. NFA selects the departure airport with most problems, quantifiable by the largest number of flights with delay (i.e., the largest HHH node) as the most prominent location for the experiment. This information is impossible to capture with HHH ordering (cf. Figure 18(a)). In HHH ordering, nodes are sorted according to the largest HHH, which in this case lies at the region grouping level (Toscany-Veneto). This implies that attribute values at the lowest level are not guaranteed to appear in order of decreasing frequency as in dataset ordering (cf. Figure 18(b)). To solve the problem of finding the



(a) VHHH: HHH Ordering (b) VHHH: Dataset Ordering

Fig. 18. VHHH: HHH Ordering and Dataset Ordering of the same Dataset

departure airport causing most delays, the NFA has just to apply the dataset ordering and look at the first attribute of the departure hierarchy (i.e., airport of Milan).

Consider now the dataset as a representation of the number of national flights per day. NFA is monitoring routes with the intent of identifying the air space in which most air-traffic is concentrated. Dataset ordering (Figure 18(b)) shows that the most frequent route is Milan-Naples. However, NFA is not interested in a frequency of a single route, but in the overall frequency of the routes independent of the grouping level. Therefore, we need the HHH ordering (cf. Figure 18(a)). In this way, we can immediately notice that most air-traffic is concentrated above regions Toscany and Veneto.

6 Conclusions and Future Work

This paper presents VHHH, a tool to visualize and analyze a dataset with the help of HHHs. The solution inputs a two-dimensional dataset, a threshold, computes HHH information for the dataset, and visualizes the result to the user. The paper proposes an interpretation alphabet and describes common patterns to ease the mining of datasets with the help of the VHHH tool. The proposed solution is based on a new count-balancing technique that provides a more accurate computation of HHH at intermediate levels of the lattice. The paper offers two orderings for categorical values that preserve the hierarchical information of the attributes. Extensive experimental evaluations of VHHH on synthetic and real world datasets show that VHHH detects anomalies, provides an overview of the data, and points to the potential weaknesses.

There are two interesting directions for future work. First, the method could be generalized to high dimensional data. High dimensional data is inherently more complex to analyze and visualize. This is an interesting challenge to develop

a simple visual analysis method that provides powerful analytics. Second, online HHH computation algorithms would be interesting to develop. Currently, the algorithms assume an input threshold to be known in advance. A re-computation of HHH information is required if the parameter changes.

References

1. Jean-Marc Adamo. *Data mining for association rules and sequential patterns: sequential and parallel algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.
2. Fabian Bendix, Robert Kosara, and Helwig Hauser. Parallel sets: Visual analysis of categorical data. In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 18, Washington, DC, USA, 2005. IEEE Computer Society.
3. Alina Beygelzimer, Chang-Shing Perng, and Sheng Ma. Fast ordering of large categorical datasets for better visualization. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–244, New York, NY, USA, 2001. ACM Press.
4. Sharma Chakravarthy and Hongen Zhang. Visualization of association rules over relational dbmss. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 922–926, New York, NY, USA, 2003. ACM Press.
5. Gouthami Chintalapani, Catherine Plaisant, and Ben Shneiderman. Extending the utility of treemaps with flexible hierarchy. *iv*, 00:335–344, 2004.
6. Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Finding hierarchical heavy hitters in data streams. In *VLDB*, pages 464–475, 2003.
7. Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Diamond in the rough: finding hierarchical heavy hitters in multi-dimensional data. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 155–166, New York, NY, USA, 2004. ACM Press.
8. Michael Friendly. Visualizing categorical data: Data, stories, and pictures. In *SAS Users Group International 25th Annual Conference*, 2000.
9. John Hershberger, Nisheeth Shrivastava, Subhash Suri, and Csaba D. Tóth. Space complexity of hierarchical heavy hitters in multi-dimensional data streams. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 338–347, New York, NY, USA, 2005. ACM Press.
10. Chien-Kuo Chen Jen-Peng Huang Huang-Cheng Kuo, Yi-Sen Lin. Mapping algorithms for indexing categorical data.
11. Erica Kolatch and Beth Weinstein. CatTrees: Dynamic visualization of categorical data using treemaps. 2001.
12. Yan Liu and Gavriel Salvendy. Visualization to facilitate association rules modelling: A review. In *Ergonomia IJEAndHF, 2005, Vol. 27, No.1*, pages 11–23, 2005.
13. S. Ma and J. Hellerstein. Ordering categorical data to improve visualization, 1999.
14. George M. Marakas. *Modern Data Warehousing, Mining, and Visualization: Core Concepts*. Pearson Education, 2002.
15. Geraldine E. Rosario, Elke A. Rundensteiner, David C. Brown, Matthew O. Ward, and Shiping Huang. Mapping nominal values to numbers for effective visualization. *Information Visualization*, 3(2):80–95, 2004.

16. Jisheng Wang, David J. Miller, and George Kesidis. Efficient mining of the multidimensional traffic cluster hierarchy for digesting, visualization, and anomaly identification. Technical Report NAS-TR-0023-2005, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, August 2005.
17. Pak Chung Wong, Paul Whitney, and Jim Thomas. Visualizing association rules for text mining. In *INFOVIS '99: Proceedings of the 1999 IEEE Symposium on Information Visualization*, page 120, Washington, DC, USA, 1999. IEEE Computer Society.
18. Yin Zhang, Sumeet Singh, Subhabrata Sen, Nick Duffield, and Carsten Lund. Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 101–114, New York, NY, USA, 2004. ACM Press.